CONSTRUCTING K-CONNECTED M-DOMINATING SETS IN WIRELESS SENSOR NETWORKS

Yiwei Wu*, Feng Wang†, My T. Thai‡ and Yingshu Li*

*Department of Computer Science,
Georgia State University, {wyw, yli}@cs.gsu.edu
†Department of Mathematical Sciences and Applied Computing,
Arizona State University, Feng.Wang.4@asu.edu
‡Department of Computer and Information Sciences and Engineering,
University of Florida, mythai@cise.ufl.edu

***Abstract*—A k-Connected m-Dominating Set (kmCDS) working as a virtual backbone in a wireless sensor network is necessary for fault tolerance and routing flexibility. In order to construct a kmCDS with the minimum size, some approximation algorithms have been proposed in the literature. However, all of those algorithms only consider some special cases where $k = 1, 2$ or $k = m$. In this paper, we propose one centralized heuristic algorithm CGA and one distributed algorithms, DDA which is deterministic, to construct a kmCDS for general k and m. Simulation results are also presented to evaluate our algorithms and the results show that our algorithms have better performances than the exiting other algorithms.**

I. INTRODUCTION

Wireless Sensor Networks (WSNs) composed of a set of static or mobile nodes are now used in many applications, including environment and habitat monitoring, traffic control, and *etc*. Since sensor nodes are tightly constrained in processing ability, storage capacity and energy, routing and data aggregation in WSNs are very challenging due to these inherent characteristics. Therefore, a robust *virtual backbone* is highly desired for a WSN. Using a virtual backbone infrastructure which is one kind of hierarchical methods is an efficient way to lower energy consumption in routing and performing data aggregation.

For a graph $G(V, E)$, a *Dominating Set* S of G is defined as a subset of V such that each node in $V \setminus S$ is adjacent to at least one node in S. A *Connected Dominating Set (CDS)* C of G is a dominating set of G which induces a connected subgraph of G. The nodes in C are called *dominators*, the others are called *dominatees*. A CDS is the earliest structure proposed as a candidate for virtual backbones in WSNs. Using this virtual backbone, a sender can send messages to its neighboring dominator. Then along the *CDS*, the messages are sent to the dominator closest to the receiver. Finally, the messages are delivered to the receiver. Furthermore, a CDS can also be organized into a hierarchy to reduce control overhead.

A CDS only preserves 1-connectivity. However, to achieve robustness, k-connectivity should be guaranteed, where k-connectivity requires that between any pair of nodes in a CDS there exist at least k different paths. With k-connectivity, communication may not be disrupted even when up to $k-1$ paths fail. k-connected virtual backbones also provide multi-path redundancy for load balancing or transmission error tolerance. Meanwhile, we should also consider fault tolerance and robustness for those dominatee nodes which are not in a CDS. Then there comes the m-domination constraint which requires that each dominatee node has at least m neighboring dominators in a CDS. Therefore, one dominatee node still can be connected with the CDS even up to its $m - 1$ dominator neighbors are dead.

In this paper, we investigate how to construct a k-Connected m-Dominating Set (kmCDS). Two algorithms are proposed to construct a kmCDS. The first one, CGA, is a centralized algorithm and the other one, DDA, is a distributed one. CGA first constructs an m-dominating set and then augments this set to be k-connected. DDA is a deterministic algorithm whose procedure is the same as CGA. Several approximation algorithms for constructing kmCDSs have been proposed in the literature. However, all these algorithms only consider some special cases where $k = 1, 2$ or $k = m$. The main contribution of our work is that we propose two algorithms to consider general k and m. Given these two algorithms, users have more options for a variety of applications with different requirements to obtain robust and fault-tolerant virtual backbones.

1-4244-1513-06/07/$25.00

The remainder of this paper is organized as follows. In Section II, we review some existing kmCDS construction algorithms. In Section III, the problem definition and the underlying wireless network model are presented. In section IV, we present CGA. Sections V is devoted to DDA. Since DDA is a deterministic algorithm, the performance ratio is also provided. The time and message complexities of these algorithms are presented as well. In Section VI, we evaluate the performances of our algorithms through simulations. Finally, we conclude this paper and discuss future research directions in Section VII.

II. Related Work

The problem of constructing a minimum CDS is NP-hard. Some centralized and distributed algorithms were proposed in [12], [1], [6], [7], [15], [9], [13], and [14]. However, none of these works investigate how to construct a kmCDS for $k > 1$ and $m > 1$.

In our work, considering the robustness and fault-tolerance requirements of WSNs, we investigate how to construct a kmCDS for general k and m. The most related works to us are [4], [5] and [8]. In [4], three localized K-CDS construction protocols were proposed. The first one is a probabilistic approach which is based on K-Gossip. The second is a deterministic approach which is an extension from the K-coverage condition. The last one is Color-Based K-CDS Construction. The major difference between our work and theirs is that their approaches only consider to construct a kmCDS where $k = m$. In [5], the algorithm Connecting Dominating Set Augmentation (CDSA) to construct a 2-connected virtual backbone was proposed. This algorithm first constructs a *CDS*, and then it computes all the blocks and adds intermediate nodes to make all the backbone nodes being in the same block. Similarly, this is also only for the case where $k = 2$ and $m = 1$. In [8], a centralized algorithm was proposed which requires the input graph to be at least $max(k, m)$ connected. Most of the existing works do not take energy into consideration, while energy should be one of the primary concerns especially for WSNs to prolong network lifetime. Our work investigates how to construct a kmCDS for general k and m. Furthermore, energy is taken into account when we construct a CDS.

III. Preliminaries and Network Model

In this paper, we are mainly interested in static symmetric multi-hop WSNs. The topology of a network is represented as a *Unit Disk Graph* (UDG), denoted as $G(V, E)$, where V is the node set and E is the edge set. The main purpose of choosing a UDG as a network model is to illustrate the performance ratio of our algorithms. Otherwise, our network model can be relaxed to general undirected graphs. We also use *nodes* and *vertices* interchangeably in the context of graph theory and WSNs. We adopt the following definitions to illustrate our work.

Definition 3.1: A graph G is said to be k*-vertex connected* or k*-connected* if for each pair of vertices there exist at least k mutually independent paths connecting them. In other words, the graph G is still connected even after removal of any $k - 1$ vertices from G.

Definition 3.2: In a graph $G(V, E)$, a vertex is said to dominate itself and all of its neighbors. An m*-dominating set* D is a set $D \subset V$ such that every vertex in $V \backslash D$ is dominated by at least m vertices in D. †

Definition 3.3: A set $C \subset V$ is a k-connected m-dominating set (kmCDS) of graph $G(V, E)$ if the induced subgraph $G'(C, E')$ is k-vertex connected and the set C is also an m-dominating set of G.

In this paper, we investigate how to construct a kmCDS for a given graph $G(V, E)$ with the assumption that such a kmCDS exists in G.

IV. Centralized Algorithm (CGA)

In this section, we present a centralized algorithm, CGA, for constructing a kmCDS. Following is a list of notations we use:

- N_i: The number of neighbors of a node i.
- e_i: The energy of a node i.
- N_i^c: The number of dominator neighbors of node i in C.
- ID_i: The ID of node i.
- C: A k-connected m-dominating set.

The main idea is to construct an m-dominating set first and then augment this set for k-connectivity. Firstly, nodes are sorted in non-increasing order based on tuple (N_i, e_i, ID_i). N_i is given the highest preference because of the observation that the size of C is smaller if nodes with larger degree are added first. Energy is another consideration. Therefore, the nodes with more remaining energy are added to the set instead of the ones with less remaining energy so that the total network lifetime can be extended. Node ID is used to break ties. Initially, C is empty. Then nodes are repeatedly added into C till C is an m-dominating set. After C becomes an m-dominating set, we need to check whether C is k-connected or not. The procedure *FindkmCDS* stops till C is k-connected. The

†Note: In some literatures, the m-dominating set D requires each node in V to be dominated by at least m nodes in D. In this paper, we do not adopt this definition because we also need to consider the k-connectivity of set D.

set C then is a kmCDS. However, this greedy algorithm cannot guarantee an optimal solution. It is possible that some redundant nodes present in C. In order to get a set C with a small size, those redundant nodes should be removed from C using the procedure *Optimization*. In the procedure *Optimization*, we firstly consider those nodes with small (N_i, e_i, ID_i), and check if one node i has m dominators in set C or not. Then we check if this node i is removed, all i's dominatee neighbors still have m dominators in C. The most important part is to check after the removal of i, C is still k-connected. After all those requirements are satisfied, i can be removed. These steps are repeated till all the nodes in C have been checked. The whole process of CGA is illustrated in Algorithm 1.

Algorithm 1 CGA($k, m, G(V, E)$)

```
1: procedure FINDKMCDS(k, m, G(V, E))
2:     Sort nodes in non-increasing order in G based on
   their (N_i, e_i, ID_i)
3:     C ← φ
4:     for i = 1 to |V| do
5:         if N_i^c < m then
6:             C ← C ∪ {v_i}
7:         end if
8:     end for
9:     while C is not k connected do
10:        add a dominatee with highest (N_i, e_i, ID_i) into
   C
11:    end while
12:    return C
13: end procedure

14: procedure OPTIMIZATION(k, m, C)
15:    Sort nodes in non-decreasing order in C based on
   their (N_i, e_i, ID_i)
16:    for i = 1 to |C| do
17:        if N_i^c ≥ m then
18:            if ∀v_j|N_j^c ≥ m after remove v_i then
19:                if C is k-connected after remove v_i
   then
20:                    C ← C − {v_i}
21:                end if
22:            end if
23:        end if
24:    end for
25:    return C
26: end procedure
```

Theorem 4.1: The set C derived from algorithm CGA is a k-connected m-dominating set.

Proof: In the outer *for* loop of *FindkmCDS*, only the nodes whose N_i^c is less than m are added to C. This procedure can guarantee that all the dominatees have at least m dominator neighbors in C. In the *while* loop, we can guarantee that C is k connected. We also assume that this graph G has at least one kmCDS. So, this procedure can obtain a correct kmCDS. The procedure of *Optimization* also preserves this property. Then the set C obtained from CGA is a k-connected m-dominating set. ■

Theorem 4.2: The time complexity of CGA is $O(|V|^{3.5} \cdot |E|)$.

Proof: In procedure *FindkmCDS*, sorting nodes takes $O(|V|log|V|)$ time. It also needs $O(|V|\Delta)$ time in the *for* loop, where Δ is the maximum node degree of the graph. By using network flow techniques [11], a test on whether two vertices are k-connected can be found in $O(|V|^{0.5} \cdot |E|)$ time. The time to test whether one vertex is k-connected to others is $O(|V|^{1.5} \cdot |E|)$. Therefore, to check whether the whole graph is k-connected or not, $O(|V|^{2.5} \cdot |E|)$ time is needed. The total running time of procedure *FindkmCDS* is $O(|V|^{3.5} \cdot |E|)$. The running time of procedure *Optimization* is $O(|V|(\Delta + \Delta^2 + |V|^{2.5} \cdot |E|))$. Therefore, the time complexity of CGA is $O(|V|^{3.5} \cdot |E|)$. ■

V. Distributed Deterministic Algorithm (DDA)

In WSNs, especially for WSNs with a large number of nodes, it is more practical to employ distributed algorithms. Therefore, we design a distributed deterministic algorithm DDA for constructing a kmCDS. The main idea of DDA is that an m-dominating set can be obtained by constructing a 1-dominating set m times followed by making this m-dominating set k-connected according to Lemma 5.1.

Lemma 5.1: If G is a k-connected graph, and G' is obtained from G by adding a new node v with at least k neighbors in G, then G' is also a k-connected graph. This Lemma was proved in [3].

We now present our deterministic algorithm which consists of three phases:

1. Using one of the distributed CDS algorithms such as [7], [9], [13] or [15] to construct a CDS C of G.
2. Using one of the distributed *Maximal Independent Set (MIS)* algorithms such as [7], [9], [13] or [15] to make C a 1-connected m-dominating set by adding $m-1$ MISs from $G \backslash C$.
3. Connecting set C for k-connectivity.

In phase 1 and phase 2 we use r-CDS [15] to construct a CDS and MISs, since it is totally localized. Therefore, we only illustrate phase 3 here.

After phase 2, all the nodes in the CDS and MISs found in phase 1 and phase 2 are black. Other nodes are white.

Every node could have one of the following three status which are *Initialization*, *Pending* and *Done*. Before phase 3, every node's status is *Initialization*. At each node, the following lists are maintained. One is *Requestor List* used to store the IDs of those nodes who request this node to join the k-*connected component*. The other is *Black Adjacent List* which is used to store the black neighbors.

Firstly, the leader who is randomly selected starts neighbor information collection. We have already collected two-hop-away neighborhood information in phase 1 and 2. Then the leader begins to build a k-connected subgraph. [†] If unsuccessful, then the leader collects three-hop-away neighborhood information and so on till a k-connected subgraph can be built. This strategy is reasonable, although we need multi-hop-away neighborhood information. If at least one kmCDS exists in the graph, we can always find this k-connected subgraph. The worst case is that we need all nodes' information.

After build a k-connected subgraph, the leader notifies all the nodes in its k-connected subgraph. Then all these nodes turn black and broadcast a *K-ConnectedComponent (KC)* message.

A node who receives at least one *KC* message is called a *candidate node*. That means this node is adjacent to the k-connected component. Only those candidate nodes process the messages they receive. Others discard the messages directly. Following are the actions each node carries out upon receiving different types of messages.

A. K-ConnectedComponent (KC) $\langle ID \rangle$ message

Upon receiving a *KC*$\langle u \rangle$ message from u, a black node x decreases k by one. If $k = 0$, this means this black node already has k neighbors in the k-connected component and can join the k-connected component. It then broadcasts a *KC* message and turns its status to *Done*. If $k > 0$ and the status is *Initialization*, then node x needs connectors to connect to the k-connected component. It adds its ID to the *Requestor List* in the *RequireConnector (RC)* message and sends this message. Then x goes into *Pending* status.

Upon receiving a *KC*$\langle u \rangle$ message, a white node x decreases k by one. If $k = 0$ and the status is *Pending* or *Done*, x removes u from the *Requestor List* and broadcasts a *ACKConnected (AC)* message to all the nodes in its *Requestor List* and sends *ConfirmUnuse (CU)* messages to ask those nodes to abort the join actions initialized by x.

[†]Note: In this paper, k-connected subgraph and k-connected component are interchangeable.

B. RequireConnector (RC) $\langle ID, RequestorList \rangle$ message

In this message, ID is the sender's ID, and *Requestor List* is used to store the IDs of nodes who request a path to join k-Connected Component so far.

Upon receiving an *RC*$\langle x, L \rangle$ message, both of white and black nodes check whether their IDs are in *L* or not. If so, these *RC* messages should be dropped.

Upon receiving an *RC*$\langle x, L \rangle$ message, a black node y of *Initialization* or *Pending* status stores x into its *Requestor List*. A black node y of *Done* status sends *AC* to u because y has already joined the k-Connected Component.

Upon receiving an *RC*$\langle x, L \rangle$ message, a white node v of *Initialization* status first checks whether $k = 0$. If so, v sends an *AC* message to x and goes to status *Done*. This means that v can join the k-Connected Component because it already has k black neighbors in the k-Connected Component. Otherwise, v adds its ID to the list L and forwards $RC\langle v, L \rangle$ and also adds x to its *Requestor List*. If the status of v is *Pending*, v adds x to its *Requestor List*. If the status of v is *Done*, v sends *AC* to x.

C. ACKConnected (AC) $\langle ID, PathList \rangle$ message

In this message, *Path List* contains all the IDs of nodes who request a path to join the k-Connected Component. Upon receiving an *AC* $\langle v, L \rangle$ message, both of the white and black nodes first check whether they appear in the list *L*. If not, then this *AC* message is dropped and k is decreased by one. If the status of the black node x is *Pending*, x sends a *ConfirmSuccess (CS)* message to v. If $k = 0$, x goes to status *Done* and broadcasts a *KC* message to notify its neighbors that it has already joined the k-Connected Component. If the status of the black node x is *Done*, x sends a *ConfirmUnuse (CU)* message to v.

If the status of a white node u is *Pending* and $k = 0$, u sends *AC* messages to all the nodes in its *Requestor List*. Then u empties this list. Upon receiving an *AC*$\langle v, L \rangle$ message, a white node of status *Done* sends a *CU* message to v.

D. NACKConnected (NC) $\langle ID, PathList \rangle$ message

If the nodes who receive a *NC* $\langle v, L \rangle$ message find their IDs are not in the list *L*, this *NC* is dropped. If a white node u receives all *NC* messages from its white neighbors, then u sends *NC* messages to all the nodes in its *Requestor List* and goes back to status *Initialization*.

E. ConfirmSuccess (CS) $\langle ID, PathList \rangle$ message

If one node receives a *CS*$\langle x, L \rangle$ message and its ID is not in the list *L*, then the *CS* message is dropped. Upon

receiving a *CS*$\langle x, L\rangle$ message, a white node v removes its ID from the list *L*. If *L* is not empty, v broadcasts this *CS* message. Then v turns to status *Done* and empty its *Requestor List*.

F. ConfirmUnuse (CU) $\langle ID, PathList\rangle$ *message*

If one node receives a *CU*$\langle x, L\rangle$ message and its ID is not in the list *L*, then the *CS* message is dropped. Upon receiving a *CU*$\langle u, L\rangle$ message, a white node v removes its ID from the list *L* and also removes u from its *Requestor List*. If this *Requestor List* is empty, then it broadcasts this *CU* message and then turns to *Initialization* status.

Lemma 5.2: Every subset of an *MIS* is at most three hops away from its complement.

Proof: We prove by contradiction. Let M be a subset of the MIS and $\overline{M}$ be the complement of M. Suppose for the purpose of contradiction that there exists a path $M \rightarrow v_1 \rightarrow v_2 \rightarrow v_3 \rightarrow \overline{M}$ in the graph G between M and $\overline{M}$, and the distance between M and $\overline{M}$ is larger than 3. Since $\overline{M}$ is the complement of M, nodes v_1, v_2 and v_3 should not be the nodes in the MIS and must be dominatees. In this situation, node v_2 is not dominated by any node in MIS. This violates the definition of MIS. Therefore, the distance between M and $\overline{M}$ is at most three hops away from each other. ■

Lemma 5.3: Let $G = (V, E)$ be any *UDG* and m be any constant such that $\delta_G \geq m - 1$ where δ_G is the minimum node degree of graph G. Let D_m^* be any optimal m-domination of G and S be any MIS of G. Then $|S| \leq \frac{5}{m}|D_m^*|$.
This Lemma has been proved in [10] which will be used to prove the performance ratio of DDA.

Theorem 5.1: The set C derived from phase 2 is a 1-connected m-dominating set.

Proof: DDA constructs one CDS C_0 in the first phase. If any dominatee is added into C_0, the new set C_0 is still connected. Let C_i be the connected dominating set after constructing i MISs, $C_i = C_{i-1} \cup M_i$, where M_i is the 1-dominating set derived from $V \backslash C_{i-1}$. Node v is an arbitrary node in V, and we now prove the theorem by induction.

1. Base case: When $i = 0$, one CDS C_0 is constructed.
2. Induction: If C_{i-1} is a 1-connected i-dominating set. Then every v has at least i neighbors in C_{i-1}. Now we need to prove that C_i is a 1-connected $(i + 1)$-dominating set after adding M_i. After constructing M_i, the nodes not in C_{i-1} and M_i has i neighbors in set C_{i-1} and one neighbor in M_i. Therefore, after adding all the nodes in M_i to C_{i-1}, the set C_i is a $(i + 1)$-dominating set.

Therefore, after construct $m - 1$ dominating sets, C_{m-1} derived from phase 1 and phase 2 is a 1-connected m-dominating set. ■

Theorem 5.2: The set C derived from DDA is a k-connected m-dominating set.

Proof: From Theorem 5.1, C is an m-dominating set. In the phase 3, we add nodes into it to obtain k-connected according to Lemma 5.1. Then the set C is a k-connected m-dominating set. ■

Theorem 5.3: The message complexity of DDA is $O(|V|\Delta^2)$ and time complexity is $O(m\Delta + Diam)$, where Δ is the maximum node degree and $Diam$ is the diameter of the network.

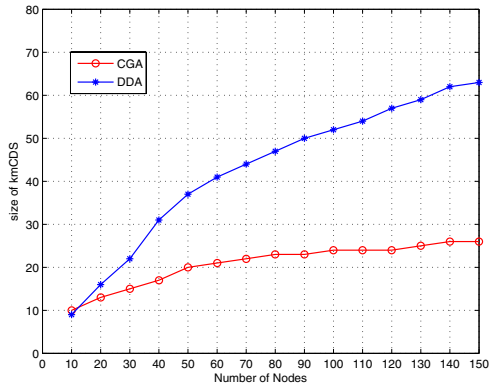
Proof: According to [15], the message complexities of phase 1 and phase 2 are $O(|V|\Delta^2)$ and $O((m-1)\Delta)$ respectively. In phase 3, for every node to decide its status, at most $O(\Delta)$ messages may be sent. Therefore, the message complexity of phase 3 is $O(|V|\Delta)$. The total message complexity of DDA is $O(|V|\Delta^2)$. In phase 1 and phase 2, the time complexity is $O(m\Delta)$. In the phase 3, the time for a candidate black node to decide its status is the time to wait enough nodes to join the k-connected component. According to Lemma 5.1, we can conclude that this time is constant. Therefore, the time complexity of phase 3 is $O(Diam)$. The total time complexity of DDA is $O(m\Delta + Diam)$. ■

Theorem 5.4: If C is a $kmCDS$ obtained by DDA, then $|C| \leq \frac{5}{m}(k^2+1)(m+42)opt$, where *opt* is the size of any optimal *kmCDS* of the network.

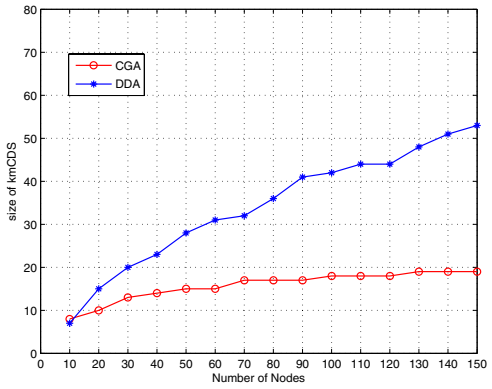
Proof: After phase 1, we have $|C_0| \leq 43|S|$ which has been proved in [15] where S is any MIS. In phase 2, $m - 1$ MISs are constructed. The number of MIS nodes constructed in this phase 2 is $(m - 1)|S|$. Therefore, The total number of black nodes in phase 1 and phase 2 is $|C| = |C_0| + (m - 1)|S| \leq (m + 42)|S|$. In the phase 3, according to Lemma 5.2 the number of nodes could be added to C is k^2 for each black node in C. After phase 3, the total number of black nodes in C is $|C| \leq (k^2 + 1)(m + 42)|S|$. According to Lemma 5.3, we have $|C| \leq \frac{5}{m}(k^2 + 1)(m + 42)opt$. Therefore, The performance ratio of DDA is $\frac{5}{m}(k^2 + 1)(m + 42)$. ■

VI. Simulations and Performance Evaluation

We conducted simulations to evaluate our algorithms. In our simulations, we randomly generated various network topologies. In order to compare with other algorithms which do not consider energy, we also assume that every node has the same energy. In each of the simulations, the nodes were placed uniformly at random within a $1000m \times 1000m$ square which also have the same trans-



(a) $k = 4, m = 3$



(b) $k = 3, m = 4$

Fig. 1: Comparison of CGA and DDA.

mission range. All of the data points were averaged over 100 simulation runs.

A. Comparison of CGA and DDA

Fig. 1 plots the size of the obtained *kmCDS* as a function of the number of nodes in a network for different k, m. When the transmission range reaches 250m, from Fig. 1(a), we observe that the size of the kmCDS obtained by CGA is the smallest, since it is a centralized algorithm. With the number of nodes increasing, the sizes of the obtained kmCDSs using those two algorithms also increase. However, the one for DDA increases much more quickly than CGA does. When the transmission range reaches 350m, from Fig. 1(b), the size of the kmCDS constructed by CGA is also the smallest. However, the sizes of kmCDSs using those two algorithms decrease when the transmission range increases. Because when transmission range increases, the network becomes dense, which means one node may have more neighbors. Therefore, a dominator can dominate more neighbors than the case when using short transmission range and the whole network would use less nodes to dominate all the nodes.

B. Comparison with other works

1) Compare CGA with CDSA [5]: Both of CGA and CDSA are centralized algorithms. We compare the size of the kmCDS generated from them. We set our simulation setting to be the same as that of *CDSA*. From Fig. 2, it is shown that CGA has a 60% improvement on the size of the obtained kmCDS on average over CDSA.

2) Compare DDA with k-Coverage [4]: We compare k-Coverage and DDA which are both distributed deterministic algorithms. The simulation setting is the same as in Section VI-B1. It is shown in Fig. 3(a) that when $k = 2, m = 2$, DDA generates a smaller kmCDS in the networks where the number of nodes is no more than 200. From Fig. 3(b), we can see when $k = 3, m = 3$, DDA has a better result than k-Coverage in the networks where the number of nodes is no more than 260.

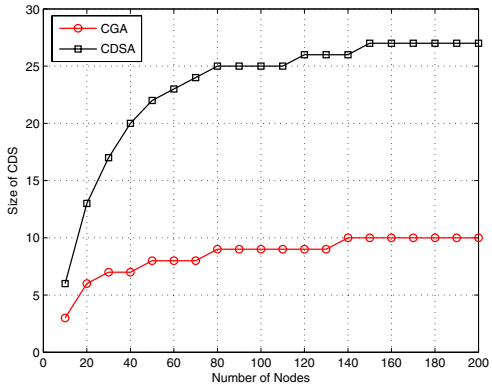
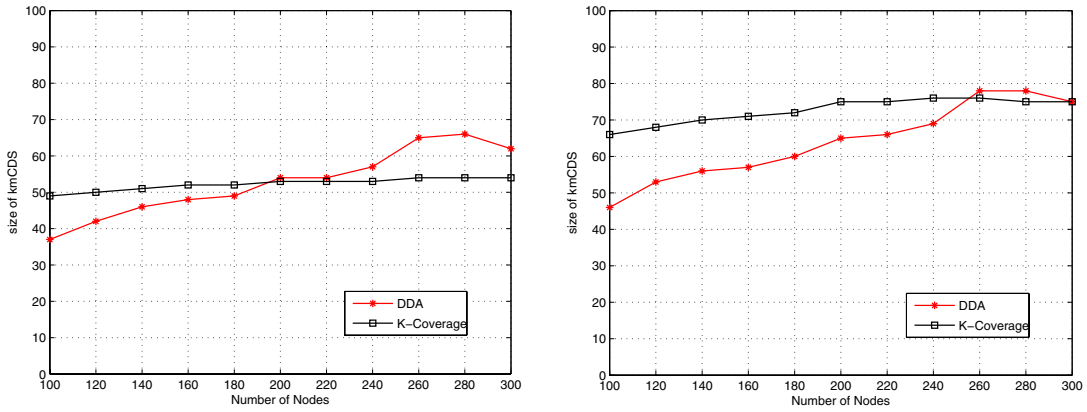


Fig. 2: Compare CGA with CDSA when $k = 2, m = 1$.

VII. Conclusion

In this paper, we investigate the problem of constructing a k-connected m-dominating set in wireless sensor networks for general k and m. We propose one centralized algorithm CGA and one distributed algorithms DDA. CGA can achieve a smaller kmCDS compared with DDA. However, DDA can be easily implemented in a real sensor network.

Our future work is to study the problem of maintaining a kmCDS in a mobility environment. Furthermore, we are interested in improving DDA since it has the overhead



(a) $k = 2, m = 2$ (b) $k = 3, m = 3$

Fig. 3: Comparision of DDA with k-Coverage.

of coordinating node activities. A probabilistic distributed algorithm with lower overhead in spite of a very low probability of failure in constructing a kmCDS could be a potential direction.

ACKNOWLEDGEMENT

This work is supported in part by the NSF CAREER Award under Grant No. CCF-0545667.

REFERENCES

[1] B. Das and V. Bharghavan, Routing in Ad Hoc Networks Using Minimum Connected Dominating Set, International Conference on Communications, 1997

[2] C. Bettstetter, On the Minimum Node Degree and Connectivity of a Wireless Multihop Network, MOBIHOC'02, 2002.

[3] D. B. West, Introduction to Graph Theory (Second Edition), Prentice-Hall, Inc. ISBN 0-13-014400-2, 2001

[4] F. Dai and J. Wu, On Constructing k-Connected k-Dominating Set in Wireless Network, IEEE International Parallel & Distributed Processing Symposium, 2005 .

[5] F. Wang, M. T. Thai, D.-Z. Du, 2-connected Virtual Backbone in Wireless Network, IEEE Transactions on Wireless Communications, accepted with revisions, 2007.

[6] J. Wu and H. Li, On Calculating Connected Dominating Set for Effecient Routing in Ad Hoc Wireless Networks, Proc. of the Third International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications, PP.7-14, August 1999.

[7] K. M. Alzoubi, P.-J. Wan and O. Frieder, Message-Optimal Connected Dominating Sets in Mobile Ad Hoc Networks, MOBIHOC, Switzerland, 2002.

[8] M. T. Thai, N. Zhang, R. Tiwari and X. Xu, On approximation algorithms of k-connected m-dominating sets in disk graphs Theoretical Computer Science, to appear, 2007.

[9] P. Wan, K.M. Alzoubi and O. Frieder, Distributed Construction of Connected Dominating Set in Wireless Ad Hoc Networks, in Preceeding of 21th Annual Joint Conference of the IEEE Computer and Communication Societies (InfoCom), 2002

[10] R. Klasing and C. Laforest, Hardness Results and Approximation Algorithms of k-tuple Domination in Graphs, Information Processing Letters, 89(2), PP.75-83, 2004.

[11] S. Even and R. E. Tarjan, Network Flow and Testing Graph Connectivity, SIAM, J. Computing, Col. 4, PP.507-518, 1975.

[12] S. Guha and S. Khuller, Approximation Algorithms for Connected Dominating sets, Algorithmica, vol.20, PP.374-387, 1998

[13] X. Cheng, M. Ding, D. Du and X. Jia, Virtual Backbone Construction in Multihop Ad Hoc Wireless Networks, Wirel. Commun. Mob. COmput. PP.183-190, 2006

[14] Y. Li, M.T. Thai, F. Wang, C.-W. Yi, P.-J. Wang and D.-Z. Du, On Greedy Construction of Connected Dominating Sets in Wireless Nwtworks, Special issue of Wireless Communications and Mobile Computing (WCMC), vol. 5, no. 88, PP.927-932, 2005.

[15] Y. Li, S. Zhu, My T. Thai, and D-Zhu Du, Localized Construction of Connected Dominating Set in Wireless Networks, NSF International Workshop on Theoretical Aspects of Wireless Ad Hoc, Sensor and Peer-to-Peer Networks (TAWN04), Chicago, June 2004.